階層を考慮した模倣学習と強化学習の組み合わせ

藤村 悠太朗[1,a)]　金子 知適[2,3,b)]

概要：近年，環境から得た報酬を用いて学習を行う，強化学習の手法が盛んに研究されている．Deep Q-Network が Atari2600 の様々なゲームで人間のプレイヤーを上回るスコアを達成したことが報告されており，より一般的なビデオゲームへの応用も期待されている．本研究は，世界的に有名なコンピュータゲームである Minecraft 上で動作する AI エージェントを研究対象とする．Minecraft のようなゲームは環境から報酬が与えられる機会が少なく，そのままでは学習が難しいという問題がある．この問題を解決するため，課題を階層的に分割することで模倣学習と強化学習を組み合わせる手法である hg-DAgger/Q で学習するエージェントでの実験を行い，その性質を検討した．

Imitation Learning and Reinforcement Learning using Hierarchical Structure

Yutaro Fujimura[1,a)]　Tomoyuki Kaneko[2,3,b)]

Abstract: Deep Q-Network (DQN) has achieved above-human performance on the domain of classic Atari2600 games and therefore DQN is expected to apply to many video games. However it is difficult for DQN to learn on games with sparse feedback, such as Minecraft. To solve this problem, we investigated the performance of hg-DAgger/Q, a framework to combine imitation learning and reinforcement learning using hierarchical structure. We demonstrate the strength of hg-DAgger/Q on a Minecraft environment.

1. はじめに

近年，環境から得た報酬を用いて学習を行う，強化学習の手法が盛んに研究されている．Mnih らが発表した Deep Q-Network (DQN) は，Atari2600 の様々なゲームで人間のプレイヤーを上回るスコアを達成した [1]．この DQN のような，深層ニューラルネットワークと強化学習を組み合わせた技術が，Atari2600 のゲームより新しく，より多くの人に親しみやすい，コンシューマー向けビデオゲームの開発に活用されることが期待される．しかし，研究者がコンシューマー向けビデオゲームでの実験環境を用意することは難しく，学術的な研究を行うのが容易ではなかった．

しかし，Johnson らは世界的に有名なコンピュータゲームである Minecraft[*1] 上で動作する AI プラットフォーム Malmo を発表した [2]．Malmo は，Minecraft 上で AI エージェントを動作させるプラットフォームで，ワールドや報酬関数を自由に設計できる．そのため，コンシューマー向けビデオゲームにおける，多様な実験環境での実験を容易に行うことが可能になった．

本研究の対象は，Minecraft 上で動作する強化学習のエージェントの作成である．この環境では，報酬が与えられる機会が少ないという性質があり，DQN での学習がそのままでは難しい．その問題点を解決するため，課題を階層的に分割し，教師あり学習の一種である模倣学習と，強化学習を組み合わせる手法である hg-DAgger/Q [3] を導入したエージェントを作成し，実験を行った．作成したエージェ

[1] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[2] 東京大学大学院情報学環
Interfaculty Initiative in Information Studies, the University of Tokyo
[3] 国立研究開発法人科学技術振興機構さきがけ
JST, PRESTO
[a)] yut-mak874@g.ecc.u-tokyo.ac.jp
[b)] kaneko@acm.org

[*1] `https://minecraft.net/` (Accessed: 2018-10-16)

ント同士の性能を比較することで，Minecraft における手法の性質や必要な工夫について評価した．

2. Minecraft について

Minecraft は，3D で表現された空間内をプレイヤーが自由に行動して，探索や建築を行うことができるサンドボックス型のゲームであり，プレイヤーに対して直接的に報酬や目的が提示されることが少ないゲームである．また，Minecraft のプレイ画面はプレイヤーキャラクターの一人称視点で与えられる．そのため AI エージェントが学習する場合においては，部分的に観測される情報から，推測することが必要な環境である．例えばその方法として，周囲の状況を記憶することが挙げられる．

Malmo を利用すると，AI エージェントが学習するための，ワールドや報酬関数を容易に実装ができる．例えば，ワールドに迷路の地形を生成したり，エージェントが Minecraft のゲーム内で特定のアイテムを取得することや，特定の場所に到達したりすることなどに対して，報酬を発生させたりする環境が容易に実装できる．

Malmo を用いた Minecraft における AI エージェントの作成の大会が，2017 年 [*2] 及び 2018 年 [*3] に開かれており，研究対象としての注目が集まっている．

3. 関連研究

3.1 強化学習

機械学習の手法の一つに強化学習がある [4]．強化学習の代表的な手法である Q 学習では，

- 状態集合 $\mathcal{S}$
- 行動集合 $\mathcal{A}$
- 環境報酬関数 $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- 遷移確率関数 $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$

を持つマルコフ決定過程 (Markov Decision Process, MDP) を考える．エージェントは状態 $s \in \mathcal{S}$ を観測し，行動 $a \in \mathcal{A}$ を方策 $\pi : \mathcal{S} \to \mathcal{A}$ によって選択する． そして，遷移確率関数 $P(s, a)$ から導かれた状態 s' を観測し，報酬 $r = R(s, a)$ を得る．エージェントは，環境から与えられる報酬の総和を最大化するために，行動価値関数 $Q(s, a)$ を学習する．

3.2 Deep Q-Network (DQN)

Deep Q-Network (DQN) [1] は，Q 学習における行動価値関数 $Q(s, a)$ の学習を深層ニューラルネットワークによって近似して学習する手法である．ニューラルネットワークの重みを θ としたときの行動価値関数を $Q(s, a; \theta)$ と書く．DQN では，次の誤差関数を最小化するように θ を繰り返し更新することによって学習を行う．

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right)^2\right] \tag{1}$$

ここで，θ_i とは，i 回更新が行われた後の重みを指す．

DQN は，Atari2600 の様々なゲームで人間よりも高いスコアを記録したが，ほとんどスコアを獲得できなかったゲームも存在し，そのようなゲームの例として Montezuma's Revenge があった．Montezuma's Revenge は，1 エピソードあたりに必要な行動の数が多く，更に特定の手順を満たさないと報酬が獲得できないため，エージェントが環境から報酬を得る機会が少ないという特徴があり，DQN による学習が困難であった．

このような難しい課題で実用的な性能を実現するためには，教師あり学習を用いる手法が考えられるが，教師データを用意するコストが大きいことや，学習のために大量の教師データが必要になるという問題がある．

3.3 階層的な構造設計

このような難しさを持つ課題に対して，階層的な構造を導入することで，効率的に学習を進める手法が提案されている [5]．3.1 節で考えた MDP を 2 段階に階層化するため，エージェントが達成するべき課題をサブゴール $g \in \mathcal{G}$ に分割する．このとき，エージェントは内部で「これから到達すべきサブゴール g を選択する」上位段階の方策 $\mu : \mathcal{S} \to \mathcal{G}$ と，「サブゴールを実行するために，行動 a を選択する」下位段階の方策 $\pi_g : \mathcal{S} \to \mathcal{A}$ を持つ．つまり，エージェントは Alg 1 のアルゴリズムで行動を行う．

Algorithm 1 階層化されたエージェント

```
repeat
    状態 s を観測する
    サブゴールを選択する: g ← μ(s)
    loop
        状態 s を観測する
        if サブゴール g を達成している then
            break
        行動 a を選択する: a ← π_g(s)
        行動 a を実行する
until エピソードが終わるまで
```

本研究では，上位方策 μ をメタコントローラーと呼び，下位方策 π_g をサブポリシーと呼ぶ．

また，Alg 1 で行動したエージェントは，それぞれのサブポリシー π_g は状態と行動の列 $\tau = (s_1, a_1, \ldots, s_T, a_T, s_T + 1)$ を生成する．この τ を下位軌跡と呼ぶことにすると，エージェントの階層的な軌跡は，$\sigma = (s_1, g_1, \tau_1, s_2, a_2, \tau_2, \ldots)$ と表すことができる．ただし，それぞれの τ_h の最後の状態は，s_{h+1} と次の下位軌跡 τ_{h+1} の最初の状態と一致する．σ から下位軌跡 τ_h を取り除いたもの，つまり状態とサブ

[*2] https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/ (Accessed: 2018-10-16)

[*3] https://www.crowdai.org/challenges/marlo-2018 (Accessed: 2018-10-16)

Algorithm 2 Hierarchically Guided DAgger/Q-learning (**hg-DAgger/Q**)

Input:
Function $\textsc{Pseudo}(s; g)$
Function $\textsc{Terminal}(s; g)$
$\epsilon_g > 0, g \in \mathcal{G}$

Initialize:
  $\mathcal{D}_{\mathrm{HI}} \leftarrow \emptyset$ and $\mathcal{D}_g, g \in \mathcal{G}$
  $Q_g, g \in \mathcal{G}$
for $t = 1, \ldots, T$ **do**
  新しい環境の状態 s を観測する
  $\sigma \leftarrow \emptyset$
  repeat
    $s_{\mathrm{HI}} \leftarrow s, g \leftarrow \mu(s)$ and initialize $\tau \leftarrow \emptyset$
    repeat
      $a \leftarrow \epsilon_g$-greedy(Q_g, s)
      a を実行し，次の状態 $\tilde{s}$ を観測する．
      $\tilde{r} \leftarrow \textsc{Pseudo}(\tilde{s}; g)$
      Q_g を更新: D_g から取得したミニバッチを用いて更新
      τ に $(s, a, \tilde{s}, \tilde{r})$ を追加
      $s \leftarrow \tilde{s}$
    until $\textsc{Terminal}(s; g)$
    σ に $(s_{\mathrm{HI}}, g, \tau)$ を追加
  until エピソードが終わるまで
  σ から τ_{FULL} と τ_{HI} を抽出する．
  if $\textsc{Inspect}_{\mathrm{FULL}}(\tau_{\mathrm{FULL}}) = \mathit{Fail}$ **then**
    $\mathcal{D}^* \leftarrow \textsc{Label}_{\mathrm{HI}}(\tau_{\mathrm{HI}})$
    for (s_h, g_h, τ) in σ **do**
      $g_h^* \leftarrow \mathcal{D}^*$
      if $g_h \neq g_h^*$ **then**
        break
      $\mathcal{D}_{g_h} \leftarrow \mathcal{D}_{g_h} \cup \tau_h$
    $\mathcal{D}_{\mathrm{HI}} \leftarrow \mathcal{D}_{\mathrm{HI}} \cup \mathcal{D}^*$
  else
    $\mathcal{D}_{g_h} \leftarrow \mathcal{D}_{g_h} \cup \tau_h$ for all $(s_h, g_h, \tau_h) \in \sigma$
  メタコントローラー μ を更新: $\mu \leftarrow \textsc{Train}(\mu, \mathcal{D}_{\mathrm{HI}})$

ゴールの列を上位軌跡と呼び，$\tau_{\mathrm{HI}} = (s_1, g_1, s_2, g_2, \ldots)$ と表すことができる．そして，すべての下位軌跡をつなげたものを τ_{FULL} と表す．

この階層的な構造設計と DQN を組み合わせて学習を行った手法として，h-DQN [6] や Hierarchical Deep RL Network (H-DRLN) [7] がある．

3.4 Hybrid Imitation and Reinforcement Learning

Le らは上位段階の方策 μ の学習を教師あり学習の一種である模倣学習で行い，下位段階の方策 π_g の学習を強化学習で行う手法 Hierarchically Guided DAgger/Q-learning (hg-DAgger/Q) を提案した [3]．アルゴリズムは Alg 2 に示す．

Alg 2 では，擬似報酬を与える関数 $\textsc{Pseudo}(s; g)$ と，状態 s においてサブゴール g が成功・失敗と問わずに終了したかどうかを判定する関数 $\textsc{Terminal}(s; g)$ を与える必要がある．[3] では擬似報酬として，状態 s においてサブゴール g が成功したかどうかを判定する関数 $\textsc{Success}(s; g)$ を基に，

$$\begin{cases} 1 & \text{if } \textsc{Success}(s; g) \\ -1 & \text{if } \neg\textsc{Success}(s; g) \text{ and } \textsc{Terminal}(s; g) \\ -\kappa & \text{それ以外} \end{cases} \quad (2)$$

と与えている．$\kappa > 0$ はできるだけ早くサブゴールを達成させるための，小さい定数である．

Algorithm 3 $\textsc{Inspect}_{\mathrm{FULL}}(\tau_{\mathrm{FULL}})$

if τ_{FULL} によって，エピソードの成功条件を満たした **then**
  return *Pass*
else
  return *Fail*

一般的な模倣学習では，教師データとして，デモ $(s_1^*, a_1^*, s_2^*, a_2^*, \ldots,)$ やラベル $\tau_{\mathrm{FULL}}^* = \{(s_1, a_1^*), (s_2, a_2^*), \ldots\}$ が必要になるが，階層構造を利用し，上位方策 μ は上位ラベル $\textsc{Label}_{\mathrm{HI}}(\tau_{\mathrm{HI}}) = \{(s_1, g_1^*), (s_2, g_2^*), \ldots\}$ による模倣学習，下位方策 π_g は強化学習によって学習することで，必要な教師データの数を減らしている．更に，階層構造を用いた学習では，$\textsc{Terminal}(s; g)$ の存在を前提するため，これを設計する段階で上位ラベル $\textsc{Label}_{\mathrm{HI}}$ のラベル付けを自動化することが可能である．

この手法により，上位段階の学習も強化学習で行う h-DQN [6] で強化学習のみを行うよりも，Montezuma's Revenge においてより高い性能を発揮するエージェントを作成することに成功したと主張している．また，Alg **?**の Q 学習の学習方法は DQN 以外の手法と取り替えることも可能である．

3.5 Deep Reccurent Q-Network (DRQN)

DQN では過去 4 フレームの画面情報を入力とするため，それ以前の情報を参照することができない．Minecraft のような一人称視点のゲームでは，4 フレームより前の情報を基に空間における自分の位置などのゲームにおける状態を判断する必要がある．このような状態の観測が部分的に行うゲームにおいては，DQN のネットワークに LSTM を導入した Deep Reccurent Q-Network (DRQN) [8] が有効であるとされている．

4. 課題設定

本研究では，Montezuma's Revenge のように，特定の手順を進めることでのみ，報酬を得てクリアすることができるような環境を，Minecarft 上で作成した．

4.1 エージェントの目的

エージェントは図 2 のような 2 部屋で構成される空間の

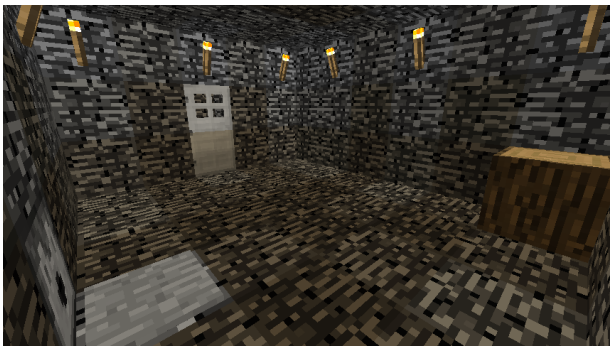

図 1　エージェントが観察できる入力例
Fig. 1

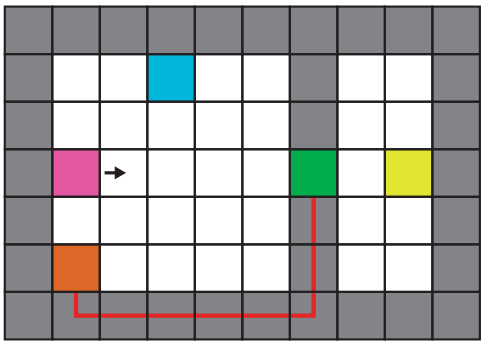

図 2　実験に用いたマップの模式図
Fig. 2

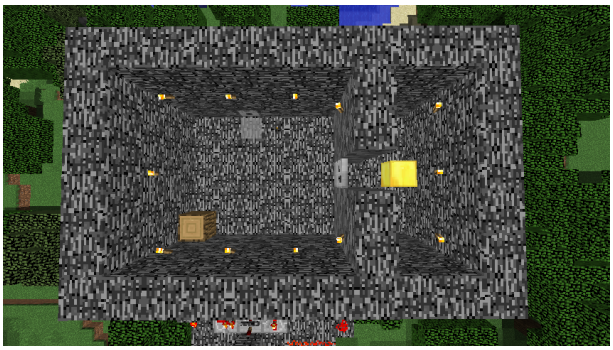

図 3　実験に用いたマップを俯瞰した様子
Fig. 3

ピンク色のマスに矢印の向きで配置される．エージェントは図 1 のような画像を 0.05 秒に 1 度の間隔で観察しながら，ゴール地点に到達することが目的である．ゴール地点である金ブロックに触れた時点で，ゲームはクリアとなりそのエピソードは終了する．Minecraft 上での実際のマップを天井ブロックを除いて上から撮影したものが図 3 である．

4.2　部屋の設計

図 2 の灰色のマスで表現された，それぞれの部屋の壁，床及び天井は岩盤ブロックで構築されていて，エージェントは破壊することができない．また，2 つの部屋の間は図 2 の緑のマスの位置にある鉄のドアで仕切られており，鉄のドアは破壊するのに非常に時間がかかるため [*4]，エージェントが時間内にゴールにたどり着くためには，エージェントが配置される最初の部屋の仕掛けを解き，鉄のドアを開けて通る必要がある．

4.3　最初の部屋の仕掛け

エージェントが最初に配置される側の部屋には，図 2 の水色のマスの位置に石の感圧板，茶色のマスに原木ブロックが配置されている．まず，エージェントが石の感圧板の上に乗ると，隣に設置されたディスペンサーからダイヤの斧が射出され，獲得することができる．次に，原木ブロックを破壊すると，隣に設置されたオブザーバーからレッドストーン回路を通じて，鉄のドアが開くという仕掛けになっている．

4.4　報酬設計

ダイヤの斧を獲得したとき，原木ブロックを破壊して獲得したとき及びゴール地点の金ブロックに触れたときに，エージェントは +1 点を獲得する．また，エージェントが行動を選択する度に −0.01 点を獲得し，ゴールにたどり着けずに 100 秒が経過した場合は −1 点を獲得する．

4.5　エージェントがとれる行動

Minecraft のゲームでは様々な行動コマンドが存在するが，本研究の実験においては以下の 7 種類に制限した．

- 何もしない．
- (前, 後) に歩く．
- 180 度 / 秒の速度で (左, 右) にカメラを回転させる．
- 攻撃を開始する
- 攻撃を終了する

5.　実験

5.1　実験概要

3.4 節で述べた hg-DAgger/Q における Q 学習を，DQN で行うエージェントを hg-DAgger/DQN とし，DRQN で行うエージェントを hg-DAgger/DRQN とする．4 章で作成した Minecraft の環境で，DQN, DRQN, hg-DAgger/DQN, hg-DAgger/DRQN のエージェントで学習を行った．実装には Python3.5 を，Minecraft での実験を行うために Malmo (0.36.0.0) を OpenAI Gym [9] の形式で取り扱えるラッパーである MarLO [*5] (0.0.1-dev16) を使用した．また，深層学習のフレームワークとして Keras (2.2.2) を用い，バックエンドは tensorflow (1.9.0) を用いた．

[*4] ピッケル以外の道具及び素手では約 25 秒
[*5] `https://github.com/crowdAI/marLo` (Accessed: 2018-10-16)

表 1　DQN のネットワーク構造
Table 1

レイヤー	パラメータ
入力	84 × 84 のグレースケール画像 4 フレーム
畳み込み層	カーネルサイズ 8 × 8 のフィルター 32 個
ReLU	
畳み込み層	カーネルサイズ 4 × 4 のフィルター 64 個
ReLU	
畳み込み層	カーネルサイズ 3 × 3 のフィルター 64 個
ReLU	
全結合層	ユニット数 512
ReLU	
全結合層	ユニット数 7
線形関数	

表 2　DRQN のネットワーク構造
Table 2

レイヤー	パラメータ
入力	84 × 84 のグレースケール画像 4 フレーム
畳み込み層	カーネルサイズ 8 × 8 のフィルター 32 個
ReLU	
畳み込み層	カーネルサイズ 4 × 4 のフィルター 64 個
ReLU	
畳み込み層	カーネルサイズ 3 × 3 のフィルター 64 個
ReLU	
LSTM	ユニット数 512
tanh	
全結合層	ユニット数 7
線形関数	

5.2　ネットワークの構造

前処理として，Minecraft から得た画像をグレースケールに変換し，84 × 84 ピクセルに縮小する．更に，Minecraft から得た画像には，獲得した斧や原木ブロックなどを含むインベントリなどの人間のプレイヤーが参照できる UI の情報が含まれていない．そのため，グレースケールの画像にインベントリにあるアイテムの情報を画像に追加した．そして得られた画像を 4 フレーム分ネットワークに入力した．

DQN のネットワークは表 1 のように [1] と同一のものを用い，DRQN のネットワークは表 2 のように [8] と同一のものを用いたが，損失の最適化には Adam [10] を使用し，学習率は 0.00025 とした．また，メタコントローラー μ の学習に用いるネットワークは [3] の実装と同一のもの，つまり表 3 のものを用い，損失の最適化は学習率は 0.00025 とした RMSProp を使用した．

5.3　ハイパーパラメータ

DQN，DRQN のハイパーパラメータは [1] の設定を利用した．ただし，同じ行動を繰り返す回数は [1] の設定では 4 回だったが，Malmo では 1 回あたりの行動間隔に差があるため，本実験では 1 回とした．

表 3　メタコントローラーのネットワーク構造
Table 3

レイヤー	パラメータ
入力	84 × 84 のグレースケール画像 4 フレーム
畳み込み層	カーネルサイズ 8 × 8 のフィルター 32 個
ReLU	
Dropout	Dropout 率 0.5
畳み込み層	カーネルサイズ 4 × 4 のフィルター 64 個
ReLU	
Dropout	Dropout 率 0.5
畳み込み層	カーネルサイズ 3 × 3 のフィルター 64 個
ReLU	
Dropout	Dropout 率 0.5
全結合層	ユニット数 512
ReLU	
Dropout	Dropout 率 0.5
全結合層	ユニット数 7
線形関数	

表 4　hg-DAgger/DQN, hg-DAgger/DRQN のハイパーパラメータの変更点
Table 4

ハイパーパラメーター	値
replay memory size	500000
target network update frequency	2000
final exploration frame	2000000
replay start size	20000

hg-DAgger/DQN，hg-DAgger/DRQN で下位段階の DQN，DRQN エージェントのハイパーパラメータは一部変更し，変更したものを表 4 に示す．

5.4　DQN,DRQN の実験結果

まず，DQN 及び DRQN を実装したエージェントの結果を図 4 に示す．学習を続けることで，徐々に環境からの報酬が僅かに増加しているが，学習に非常に時間がかかっていることが読み取れる．また，DRQN によって部分観測による難しさを克服することを期待していたが，図 4 からは DQN と比較すると大きな改善は見られなかった．

以上より，本実験に用いた実験環境は DQN や DRQN をそのまま適用するのは難しい問題であると言える．

5.5　hg-DAgger/DQN, hg-DAgger/DRQN の結果

hg-DAgger/DQN, hg-DAgger/DRQN でも同じ環境を用いて実験を行った．実験を行うにあたって，4.4 節で説明した報酬の設計でサブゴールに分割した．つまり，

サブゴール 0　「ダイヤの斧を獲得する」
サブゴール 1　「原木ブロックを獲得する」
サブゴール 2　「ゴール地点に到達する」

のようにサブゴールを設定した．また，[3] の実験で用いら

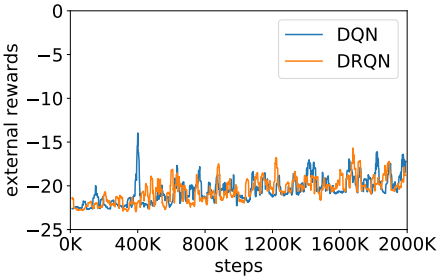

図 4　DQN, DRQN の獲得報酬

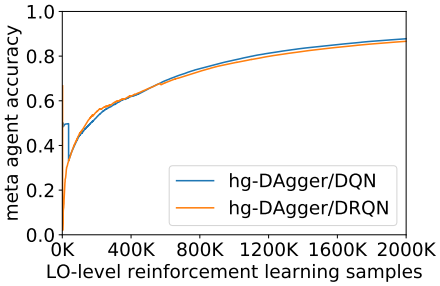

図 5　メタコントローラーの正答率

れた，メタコントローラーが誤った判断をした時点でエピソードを途中で打ち切るという工夫も実装した．

まず，メタコントローラーのサブゴール選択の正答率を図 5 に示す．ここで正答率とは，メタコントローラーが次に進むサブゴールを選択した回数に対する，適切なサブゴールを選択した回数の割合である．hg-DAgger/DQN，hg-DAgger/DRQN どちらもメタコントローラーの実装は同一であるが，図 5 も同様に学習が進んでいることが読み取れる．

一方，それぞれのサブゴールの成功率を図 6 と図 7 に示す．どちらのエージェントでも，サブゴール 0 は 100% に近い確率で成功しているが，サブゴール 1 は成功率が 30% 程度で学習が進んでいない．サブゴール 2 は最終的な成功率は 20% 程度であるが，学習が止まっている様子は読み取れない．図 8 と図 9 は，それぞれのサブゴールを担当するエージェントが行動した回数を表すグラフである．図 6 と図 8，図 7 と図 9 を比較すると，サブゴール 1 は十分な行動回数があったにもかかわらずうまく学習が進んでおらず，サブゴール 2 は行動できた回数が少なかったために，学習がまだ終了していないことがわかる．

また，サブゴールを達成した場合の，サブゴール達成のために行動した回数を図 10 と図 11 に示す．サブゴール 0 は学習が進むにつれて消費ターン数が短くなっている一方で，サブゴール 1 は消費ターン数が安定しておらず，うまく学習ができていない様子が読み取れる．また，サブゴール 1 が約 1000 回の行動を行うため，制限時間の半分程を消費してしまう．そのため，サブゴール 2 がサブゴールを達成するまでにエピソードの制限時間が終了してしまい，正の報酬を得るのが難しくなっていることが考えられる．

6. おわりに

本研究では，Minecraft で報酬を得る機会が少なく，単純な DQN や DRQN では学習が難しい環境を作成した．その環境で，階層構造を用いて模倣学習と強化学習を組み合わせて学習する hg-DAgger/Q のエージェントによる学習を行った．

実験の結果，メタコントローラーの学習は Minecraft においても正常に行うことができた．しかし，用意した実験環境のサブゴールの分割では DQN や DRQN で学習を行うにはまだ難しく，サブポリシーの学習能力に合わせて適切なサブゴール分割を行わなければ，階層構造を用いても学習が困難になることがわかった．サブゴール 1 の学習が進まなかった理由としては，原木ブロックを破壊して獲得するまでの一連の行動がランダム行動で発生しづらいため，Experience Replay からサンプルされにくかったことが考えられる．

今後の課題としては，Prioritized Experience Replay [11] などの手法を用いて，報酬が得られた経験を学習しやすくする工夫を加えることがまず挙げられる．また，サブポリシーの学習能力に合わせた適切な分割を行うための手法を導入することができれば，学習を効率よく進めることが可能になると考えられる．

謝辞　この研究の一部は，JSPS 科研費 16H02927 と JST さきがけの支援を受けています。

参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D.: Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, pp. 529–533 (online), available from ⟨http://dx.doi.org/10.1038/nature14236⟩ (2015).

[2] Johnson, M., Hofmann, K., Hutton, T. and Bignell, D.: The malmo platform for artificial intelligence experimentation, *IJCAI International Joint Conference on Artificial Intelligence*, Vol. 2016-Janua, pp. 4246–4247 (2016).

[3] Le, H., Jiang, N., Agarwal, A., Dudik, M., Yue, Y. and Daumé, III, H.: Hierarchical Imitation and Reinforcement Learning, *Proceedings of the 35th International Conference on Machine Learning* (Dy, J. and Krause, A., eds.), Proceedings of Machine Learning Research, Vol. 80, Stockholmsmssan, Stockholm Sweden, PMLR, pp. 2923–2932 (online), available from ⟨http://proceedings.mlr.press/v80/le18a.html⟩ (2018).

[4] Sutton, R. S., Precup, D. and Singh, S.: Intra-option learning about temporally abstract actions, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pp. 556–564 (1998).

[5] Sutton, R. S., Precup, D. and Singh, S.: Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning, *Artif. Intell.*, Vol. 112, No. 1-2, pp. 181–211 (online), DOI: 10.1016/S0004-

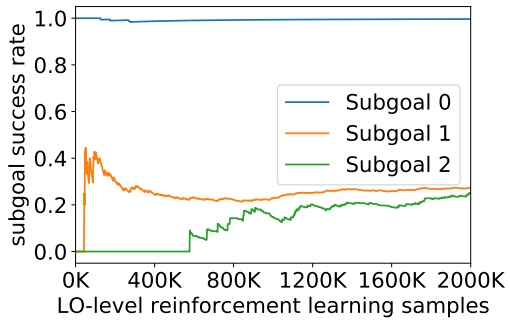

図 6　hg-DAgger/DQN のサブゴールの成功率

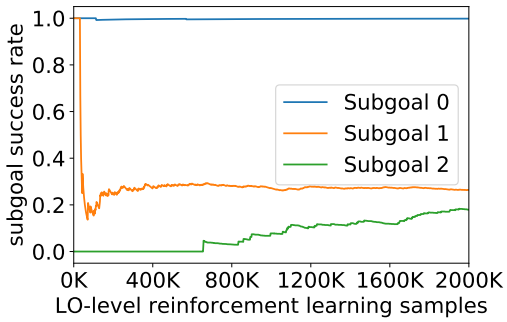

図 7　hg-DAgger/DRQN のサブゴールの成功率

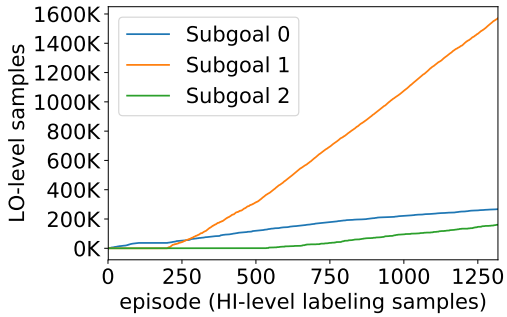

図 8　hg-DAgger/DQN のサブポリシーが行動した回数

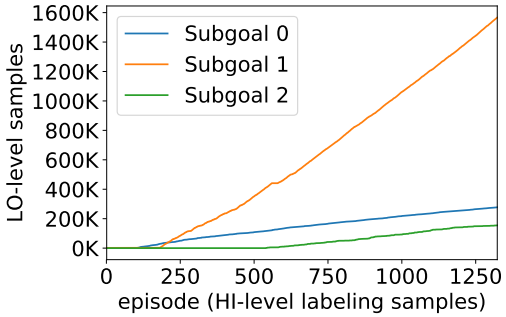

図 9　hg-DAgger/DRQN のサブポリシーが行動した回数

3702(99)00052-1 (1999).

[6] Kulkarni, T. D., Narasimhan, K. R., Saeedi, A. and Tenenbaum, J. B.: Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation, No. Nips (online), DOI: 10.1023/A:1025696116075 (2016).

[7] Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J. and Mannor, S.: A Deep Hierarchical Approach to Lifelong Learning in Minecraft., *AAAI*, Vol. 3, p. 6 (2017).

[8] Hausknecht, M. and Stone, P.: Deep Recurrent Q-Learning for Partially Observable MDPs, (online), DOI: 10.1.1.696.1421 (2015).

[9] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W.: OpenAI Gym, pp. 1–4 (online), available from ⟨http://arxiv.org/abs/1606.01540⟩ (2016).

[10] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, pp. 1–15 (online), available from ⟨http://arxiv.org/abs/1412.6980⟩ (2014).

[11] Schaul, T., Quan, J., Antonoglou, I. and Silver, D.: Prioritized Experience Replay, *CoRR*, Vol. abs/1511.05952 (online), available from ⟨http://arxiv.org/abs/1511.05952⟩ (2015).

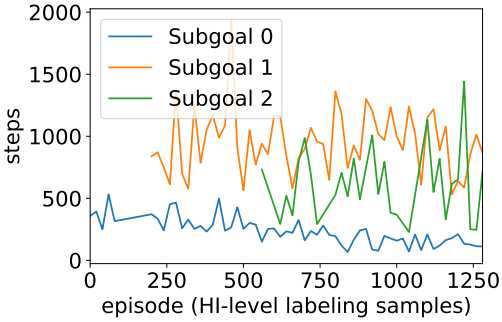

図 **10** hg-DAgger/DQN のサブゴール達成のために行動した回数

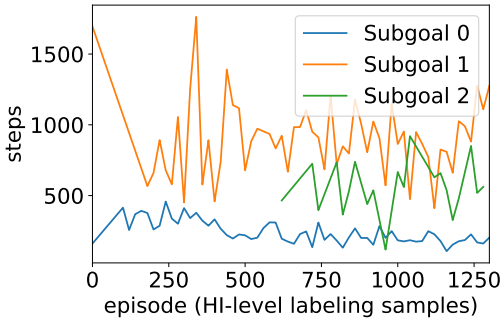

図 **11** hg-DAgger/DRQN のサブゴール達成のために行動した回数